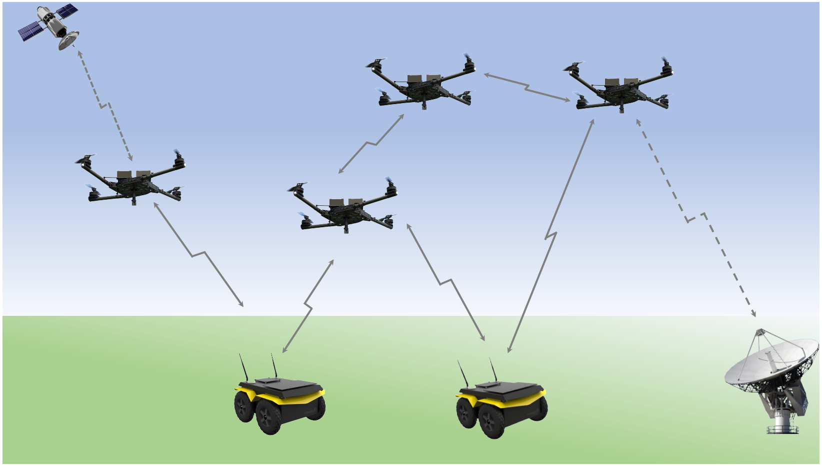
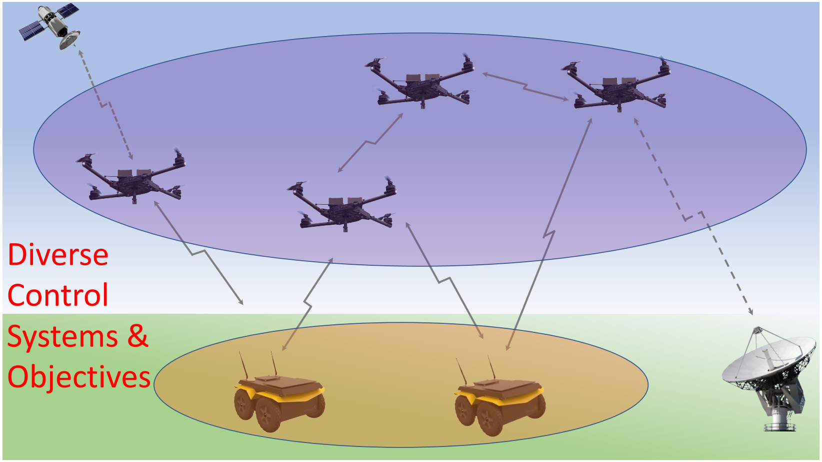
Distributed Tracking and Sensor Fusion in GPS-denied Environments

Caleb M. Bowyer and John M. Shea

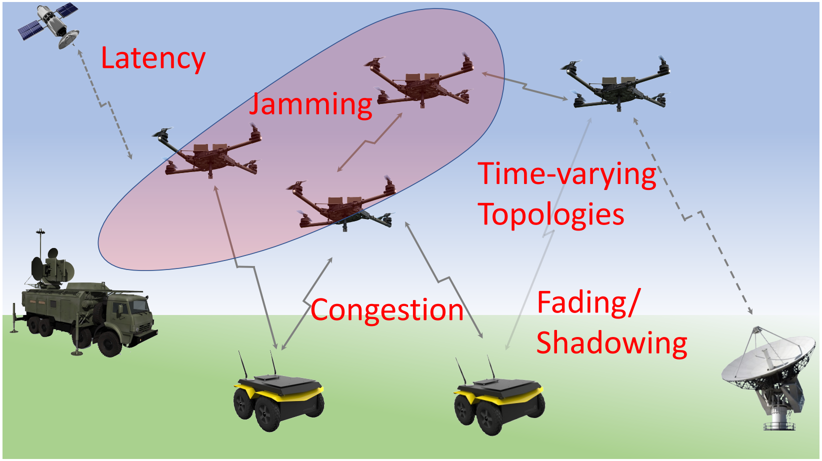
Big Picture: Joint Optimization of Control and Networks



Challenges

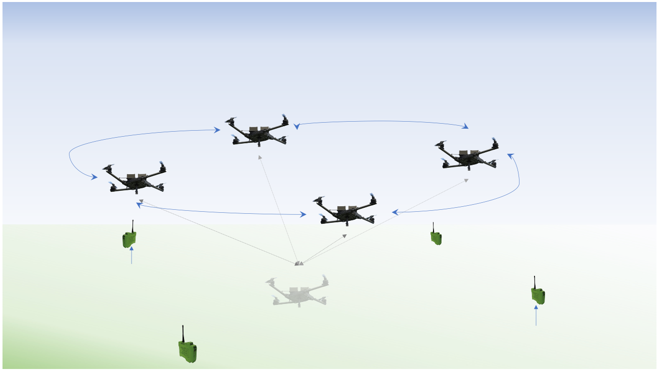


Challenges


Latency
Jamming
Time-varying Topologies
Congestion
Fading/ Shadowing

Application: Distributed Localization in GPS-Denied Environments

- Desire low cost, low complexity, robust, high-performance solutions to tracking/RADAR in GPS-denied environments



Application: Distributed Localization in GPS-Denied Environments

Application: Distributed Localization in GPS-Denied Environments

- **Low cost, low complexity:** sensors have unreliable clocks and noisy RF

Application: Distributed Localization in GPS-Denied Environments

- **Low cost, low complexity:** sensors have unreliable clocks and noisy RF
- **Robust:** no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors

Application: Distributed Localization in GPS-Denied Environments

- **Low cost, low complexity:** sensors have unreliable clocks and noisy RF
- **Robust:** no single point of failure $\Rightarrow$ distributed sensors with robustness to failure of individual sensors
- **High-performance:** need to generate reliable localization estimates using noisy ToF measurements

Need for Synchronization

- Given accurate sensor locations, tightly synchronized clocks, and perfect RF channels, distributed sensor networks can produce accurate location estimates
- Timing drifts and RF noise reduce localization accuracy
- Can reduce timing noise by using RF channel to synchronize clocks at expense of not being able to perform localization when synchronizing

Need for Synchronization

- Given accurate sensor locations, tightly synchronized clocks, and perfect RF channels, distributed sensor networks can produce accurate location estimates
- Timing drifts and RF noise reduce localization accuracy
- Can reduce timing noise by using RF channel to synchronize clocks at expense of not being able to perform localization when synchronizing

Need to optimize between localization and synchronization

System Model

- Single asset to be tracked:
  - Asset transmits beacon signal at known times to agents to facilitate tracking in GPS-denied environment
  - Asset moves according to known Markov model
- Network of m stationary sensing agents
  - Sensors measure time-of-flights (ToFs) of beacon signal & fuse measurements to localize asset

System Model – cont.

- ToFs are triply stochastic, depending on:
  - Vehicle motion state
  - Clock drift among sensors, **may be correlated over time**
  - **RF noise and received SNR depend on vehicle motion state (propagation distance)**
- Agents can synchronize (SYNCH) clocks at expense of not being able to measure ToFs during that time

Notation

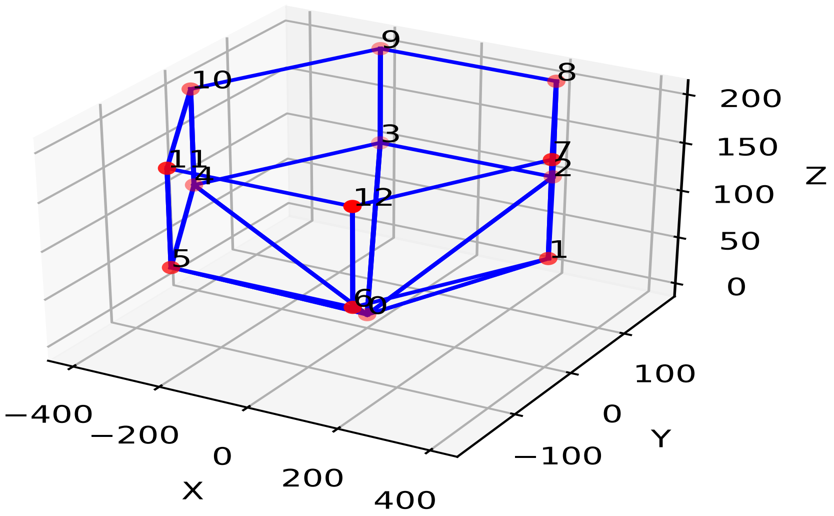
- Let $\mathbf{M}_k$ denote the motion state of asset (position) and $\mathbf{s}_k^{(i)}$ the sensor position, at time k,
- ToF from the asset to sensor i: $\hat{\tau}_{k,i} = \frac{||\mathbf{s}_k^{(i)} - \mathbf{m}_k||}{c} + N_{k,i}$
- $N_{k,i}$ is noise at time k and sensor i modeled as conditionally Gaussian with variance that depends on time since last sync T_k^s and $\mathbf{m}_k$

Variance Model

- Let σ_N^2 denote the sensor clock noise variance one slot after synchronization (i.e., when $T_k^s = 1$)
- Let σ_D^2 be the variance from RF noise when the propagation distance is one unit.
- Then, the overall noise is modeled as:

$$\begin{aligned}\left[\sigma_k^{(i)}\right]^2 &= \left[\sigma_{k,S}^{(i)}(T_k^s)\right]^2 + \left[\sigma_{k,D}^{(i)}(d_{ia})\right]^2 \\ &= T_k^s \sigma_N^2 + d_{k,i}^2 \sigma_D^2.\end{aligned}$$

Simulation: 3D Ellipse Motion Model

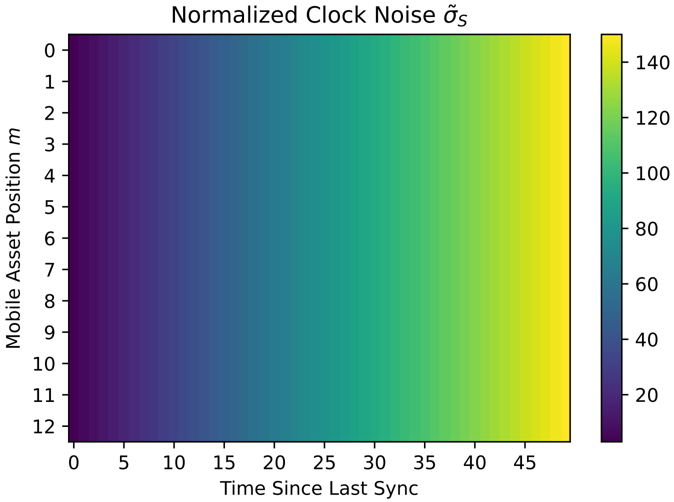


3D Ellipse Motion Model Parameters

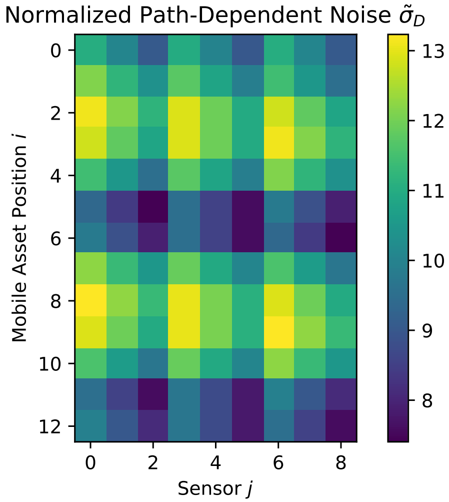
Table: Ellipse Motion Model Parameters

Parameter	**Value**
# Levels (including ground station)	3
# Levels above ground	2
Length of major axis of elliptical path	800 m
Length of minor axis of elliptical path	400 m
# States at each level above ground	6
Vertical spacing	100 m

Normalized Clock Noise $\sigma_N = 10$



Normalized RF Noise



Optimization of Localization and Synchronization

- Synchronization is required to avoid clock variance growing without bound
- Do not want to synchronize too often because not able to localize using ToFs during synchronization times
- May be able to tolerate higher timing noise when RF SNRs are high (asset is close to sensors)

Optimization of Localization and Synchronization

- Synchronization is required to avoid clock variance growing without bound
- Do not want to synchronize too often because not able to localize using ToFs during synchronization times
- May be able to tolerate higher timing noise when RF SNRs are high (asset is close to sensors)
    - However, a high SNR at some time may not reflect ambiguity built up from times when asset was operating further from sensors

Optimization of Localization and Synchronization

- Synchronization is required to avoid clock variance growing without bound
- Do not want to synchronize too often because not able to localize using ToFs during synchronization times
- May be able to tolerate higher timing noise when RF SNRs are high (asset is close to sensors)
  - However, a high SNR at some time may not reflect ambiguity built up from times when asset was operating further from sensors

Want optimal approach to fuse all available information to determine whether to synchronize or localize

POMDP

- Optimal tracking and SYNC/LOC decision can be solved using Bayesian framework:

Partially Observable Markov Decision Process (POMDP)

POMDP Formulation

- State: $(\mathbf{M}_k, T_k^s)$
- Controls: $\mathcal{U} = \{\text{sync}, \text{loc}\}$
- Continuous observations: $\hat{\boldsymbol{\tau}}_k$

  $\ldots$
- Squared-error localization cost function,

$$c_k = \left\| \widehat{\mathbf{M}}_k - \mathbf{M}_k \right\|^2$$

Beliefs

- Beliefs are *a posteriori* probabilities of states given sequence of observations and controls:

$$b_k(j) = \Pr\left(\mathbf{M}_k = j \,\middle|\, \boldsymbol{\tau}_{[0\ldots k]}, \mathbf{u}_{[0\ldots k]}\right).$$

- MAP location estimate is maximum over beliefs

$$\widehat{\mathbf{M}}_k = \arg\max_{m\in\mathcal{M}} \mathbf{b}_k(m).$$

Belief Update after Localization

- For a localization control, the Bayesian belief update at time $k+1$ is

$$b_{k+1}(j) = \eta f(\hat{\boldsymbol{\tau}}_{k+1} | j, u_k) b_k(j),$$

- where η is a normalization constant
- For a synchronization control, belief update is via model knowledge,

$$\mathbf{b}_{k+1} = \mathbf{P}^T \mathbf{b}_k$$

Solving the POMDP

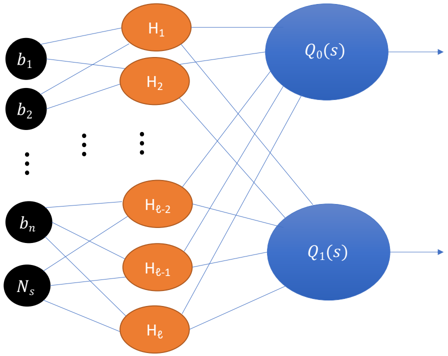
1. Can treat POMDP as an MDP where the state is now the belief state (continuous state space with many dimensions)
2. Can solve via Q-learning, but any solution approach requires approximation:
   - Quantize beliefs to create discrete space: Triple-Q learning
   - Approximate Q function as linear function of beliefs: Replicated Q-learning
   - **Approximate Q function using deep neural network: Deep Q-Learning**

Fixed Rate Deterministic (FRD)

- One more approximation: drop the continuous features and only use time since last synch:
- Fixed-rate deterministic (FRD): sync every k intervals
- Standard approach used in most of the literature, synchronization interval often chosen using ad hoc approach
- We optimize the synchronization interval to minimize the MSE
- Results still use Bayesian location estimate using ToFs and Markov movement model

DQN Architecture

- The basic DQN architecture is shown below (with unquantized belief and time since last sync as input):



DQN Architecture - Cont.

- Optimize number of layers & number of hidden neurons per layer
  - Results show no significant performance gain with more than 1 layer, more than 64 neurons in hidden layer
- RELU activation for all hidden neurons,
- Optimal policy is $u^* = \arg\min_{u \in \mathcal{U}} Q_u(s)$

Deep Q-Networks DQNs - Training

- Discount factor, $\gamma = 0.99$
- Used multiple techniques from literature to improve convergence, along with hyperparameter tuning:
  - Learning rate, $\alpha = 0.1$
  - Used *soft update*:
    - target network only updated every 100 steps
    - target network update weight $\tau = 10^{-3}$
  - Used experience replay buffer:
    - 10^6 entries
    - 64 batch size

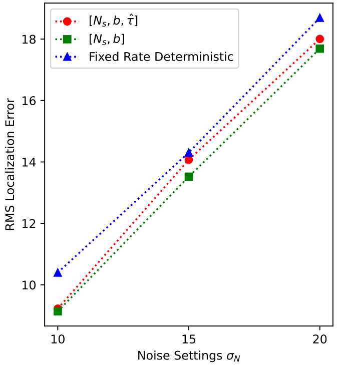
Deep Q-Networks DQNs – Training 2

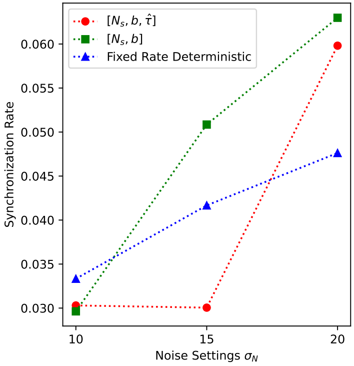
- ϵ-greedy annealed from 1 to 10^{-4} over course of training
- Standard ϵ-greedy chooses uniformly among actions
    - resulted in no learning in our application because `sync` was often chosen by ϵ-greedy before ever reaching state when `sync` is optimal action
    - used domain knowledge to make ϵ-greedy randomly choose an action that is biased to select `loc` with high probability (0.99)

DQN Features Tested

- Tested all combinations of time since last sync (N_s), time of flights ($\hat{\boldsymbol{\tau}}$), and beliefs ($\mathbf{b}$)
  1) $[N_s]$ (equivalent to FRD)
  2) $[\hat{\boldsymbol{\tau}}]$ 3) $[\boldsymbol{b}]$ 4) $[\hat{\boldsymbol{\tau}}, \boldsymbol{b}]$
  5) $[\hat{\boldsymbol{\tau}}, N_s]$ 6) $[\boldsymbol{b}, N_s]$ 7) $[\hat{\boldsymbol{\tau}}, \boldsymbol{b}, N_s]$
- Found N_s to be essential: performances of feature combos without N_s were much worse than others and are not shown

Test Performance: RMS Localization Error and Average Sync Rate

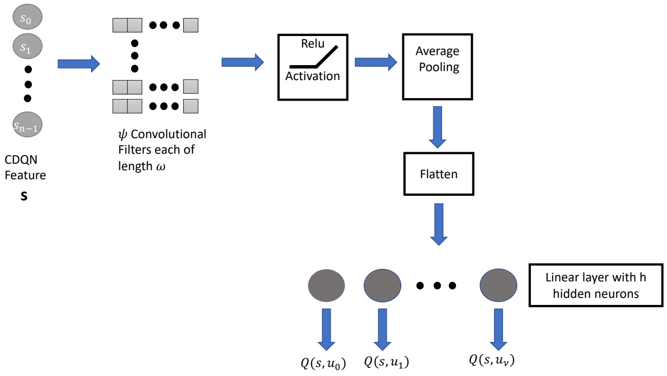




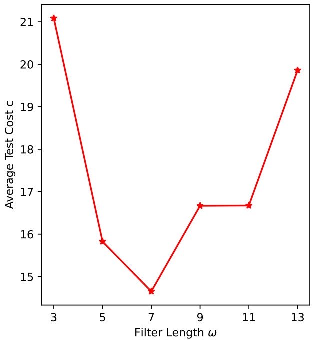
Ongoing Research – Part I: Clock Drift Model

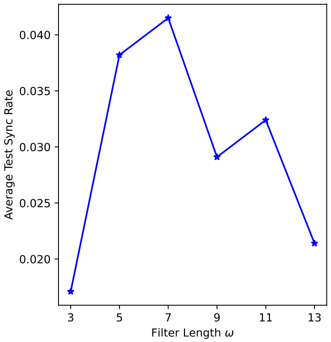
- Develop optimal synchronization/localization strategies when clock drift modeled by random walk (breaks Markov property)
- Approaches under investigation:
  1. redefine underlying MDP using pairs of states and the observation as the difference in ToFs across those states
  2. use convolutional NN to directly extract information from the beliefs in the presence of the memory effect of the random walk

Convolutional Deep Q-Network (CDQN) design



Optimizing Filter Length

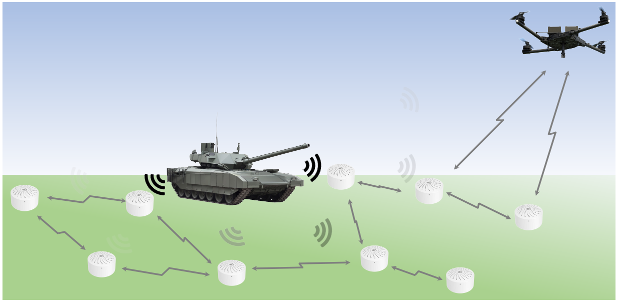




- Initial experiments show that a CNN can achieve good performance on this channel
- Filter length is much greater than the channel memory
- Average pooling works better than max pooling

Ongoing Research – Part II: Coordination of Communication for Distributed Sensors

- Distributed sensors generate noisy measurements that depend on the distance from a vehicle being tracked
- These measurements are selectively sent to a centralized fusion center over a shared wireless channel



Ongoing Research – Part II: Coordination of Communication for Distributed Sensors

- If more than one sensor transmits simultaneously, collision occurs and no sensor measurements will reach the fusion center in that interval
- Overall problem is Decentralized POMDP (Dec-POMDP)
- Dec-POMDPs are even harder to solve than POMDPs
- Can make Dec-POMDP closer to POMDP by sharing beliefs from fusion center back to agents

Ongoing Research – Part III: Optimizing Drone Data Collection and Delivery

Joint project with researchers from Pontifical Catholic University of Rio de Janeiro (PUC-RIO)

- Ground sensors are deployed to monitor a remote area but do not have long-range communication
- Multiple UAVs are tasked with collecting data from the ground sensors for delivery to a fusion center

Ongoing Research – Part III: Optimizing Drone Data Collection and Delivery

- UAVs must physically move closer to other agents to:
  - receive information from sensors
  - exchange information with other drone
  - deliver information to an internet backhaul to the fusion center
- Using Multi-agent Reinforcement Learning (MARL) to control drone movement patterns to minimize data latency

Thank You!

Any questions?